

Searching PAJ

1/2 ページ

PATENT ABSTRACTS OF JAPAN

(11)Publication number : 03-218534

(43)Date of publication of application : 26.09.1991

(51)Int.Cl.

G06F 9/46
G06F 15/16

(21)Application number : 02-183942

(71)Applicant : HITACHI LTD

(22)Date of filing : 13.07.1990

(72)Inventor : NEGISHI KAZUYOSHI
FUJII TETSUHIKO
YONEDA SHIGERU
SATO KAZUHIRO
HARADA AKIRA

(30)Priority

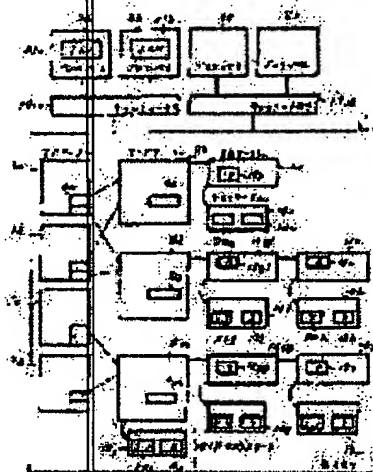
Priority number : 01185293
01304236Priority date : 17.07.1989
21.11.1989Priority country : JP
JP

(54) TASK SCHEDULE SYSTEM

(57)Abstract:

PURPOSE: To efficiently use hardware corresponding to a processors by providing virtual processor and permitting the processors to correspond to tasks when hardware that can be used only from a specified processor exists.

CONSTITUTION: When hardware 19ab which can be used only from the specified processor or the groups 2a and 2b exists, the virtual processors are provided, the specified processors 2a and 2b and the tasks are caused to correspond to the virtual processors and the queue of the tasks waiting for execution is provided corresponding to the virtual processors. The specified processors 2a and 2b take out the tasks from the queue of the corresponding virtual processors and start the execution of the tasks. Thus, the possibility of re-use when data which is executed previous time remains on a cache memory 19ab improves, for example, and task search overhead is reduced since respective processors 2a and 2b search only the tasks which can be executed.



LEGAL STATUS

[Date of request for examination]

[Date of sending the examiner's decision of rejection]

[Kind of final disposal of application other than the examiner's decision of rejection or application converted registration]

BEST AVAILABLE COPY

⑤ 日本国特許庁(JP)

⑩ 特許出願公開

⑥ 公開特許公報(A)

平3-218534

⑦ Int. Cl.⁸

識別記号

庁内整理番号

⑨ 公開 平成3年(1991)9月26日

G 06 F 9/48
15/163 6 0 B
4 3 08945-5B
6945-5B

審査請求 未請求 請求項の数 17 (全16頁)

④ 発明の名称 タスクスケジュール方式

⑪ 特 願 平2-183942

⑫ 出 願 平2(1990)7月13日

優先権主張 ⑬ 平1(1989)7月17日 ⑭ 日本(JP) ⑮ 特願 平1-185293

⑯ 発 明 者 根 岸 和 義 神奈川県川崎市麻生区王禅寺1088番地 株式会社日立製作所システム開発研究所内

⑰ 発 明 者 藤 井 哲 彦 神奈川県川崎市麻生区王禅寺1088番地 株式会社日立製作所システム開発研究所内

⑱ 発 明 者 米 田 茂 神奈川県川崎市麻生区王禅寺1088番地 株式会社日立製作所システム開発研究所内

⑲ 出 願 人 株式会社日立製作所 東京都千代田区神田駿河台4丁目6番地

⑳ 代 理 人 弁理士 小川 勝男 外1名

最終頁に続く

明 細 書

1. 発明の名称

タスクスケジュール方式

2. 特許請求の範囲

1. 複数のプロセッサが主メモリを共用し、複数のタスクを並行して実行する高多重度結合プロセッサシステムにおいて、特定のプロセッサまたはそのグループのみから利用可能なハードウェアが存在する場合に、仮想的なプロセッサを設けて、前記特定のプロセッサおよびタスクを前記仮想的なプロセッサに対応付けるとともに、該仮想的なプロセッサに対応して実行待ちタスクのキューを設けて、前記特定のプロセッサは、対応する前記仮想的なプロセッサのキューからタスクを取出し、タスクの実行を開始することとを特徴とするタスクスケジュール方式。

2. 前記特定のプロセッサまたはそのグループのみから利用可能なハードウェアを使用するタスクを、当該ハードウェアに対応する前記仮想的なプロセッサに対応付けて処理させることを特

徴とする請求項1記載のタスクスケジュール方式。

3. 前記複数のプロセッサと主メモリとの間に、複数段階のキャッシュメモリを有する場合、ある段階のキャッシュメモリを共用するプロセッサにより前記仮想的なプロセッサを構成し、各プロセッサは、対応する前記仮想的なプロセッサに実行可能なタスクが存在しない場合、他の段階のキャッシュメモリを共用するプロセッサに対応する、前記プロセッサが対応付けている仮想的なプロセッサ以外の仮想的なプロセッサの処理待ちタスクを探して処理することとを特徴とする請求項1記載のタスクスケジュール方式。

4. 前記仮想的なプロセッサと前記特定のプロセッサとの対応を、負荷の状態により定期的に変更することとを特徴とする請求項1記載のタスクスケジュール方式。

5. タスクに対して対応する前記仮想的なプロセッサの候補を指定する手段を設けて、当初候補

特開平3-218534 (2)

- となっている前記仮想的なプロセッサのいずれかでタスクの実行を開始し、以後、同一の前記仮想的なプロセッサに対応付けて処理を行うことを特徴とする請求項1記載のタスクスケジュール方式。
6. 各プロセッサ毎に、対応する前記仮想的なプロセッサの処理順序を指定する手段を設けたことを特徴とする請求項1記載のタスクスケジュール方式。
7. 高速応答を要求されるタスクに対応する前記仮想的なプロセッサを設けて、該仮想的なプロセッサに対応するプロセッサは、当該仮想的なプロセッサによる処理を優先することを特徴とする請求項1記載のタスクスケジュール方式。
8. 前記特定のプロセッサおよび前記仮想的なプロセッサに対応するテーブルを設けて、前記特定のプロセッサに対応するテーブルは対応する前記仮想的なプロセッサへのポインタのリストを有し、前記仮想的なプロセッサからタスクの制御ブロックをチェインする構造により、実行

- タスクのキューを設けるかわりに、各タスク毎に対応する仮想的なプロセッサを表示したタスクテーブルを設け、前記特定なプロセッサは、各タスクのタスクテーブルにより自プロセッサに対応する仮想的なプロセッサに対応付けられたタスクを選択し、実行を開始することを特徴とする請求項1記載のタスクスケジュール方式。
13. 前記仮想的なプロセッサは、全てのプロセッサで処理されるタスクに対応する仮想的なプロセッサであることを特徴とする請求項12記載のタスクスケジュール方式。
14. 前記ハードウェアはキャッシュメモリであることを特徴とする請求項12記載のタスクスケジュール方式。
15. 前記複数のプロセッサは主メモリを共用するとともに、複数のアドレス空間を有し、少なくとも前記アドレス空間の一つで複数のタスクを並行して実行することを特徴とする請求項12記載のタスクスケジュール方式。
16. 前記アドレス空間に対応する空間テーブルを

- 可能なタスクのキューを構成することを特徴とする請求項1記載のタスクスケジュール方式。
9. 前記プロセッサシステムの操作を行うための端末から入力された指示により、前記特定のプロセッサと前記仮想的なプロセッサとの関係および前記仮想的なプロセッサとタスクとの関係を前記端末に出力することを特徴とする請求項1記載のタスクスケジュール方式。
10. 前記複数のプロセッサが主メモリを共用するとともに、複数のアドレス空間を有し、少なくとも前記各アドレス空間の一つで、複数のタスクを並行して実行することを特徴とする請求項1～8のいずれかに記載のタスクスケジュール方式。
11. 複数のタスクが前記アドレス空間に対応付けられて処理される場合に、前記タスクの代りにタスクに対応する前記アドレス空間を、前記仮想的なプロセッサに対応付けることを特徴とする請求項1記載のタスクスケジュール方式。
12. 前記仮想的なプロセッサに対応して実行待ち

設け、各仮想的なプロセッサに対応する仕事が該アドレス空間にあることを表示する手段を該空間テーブルに設け、タスクスケジュール時に該表示手段を参照することによりタスクを実行することを特徴とする請求項15記載のタスクスケジュール方式。

17. 前記特定の仮想的なプロセッサに対応付けられたプロセッサまたはそのグループが故障した場合、それ以外の故障していないプロセッサまたはそのグループを、新たに前記仮想的なプロセッサに対応付けることを特徴とする請求項14に記載のタスクスケジュール方式。

3. 発明の詳細な説明

〔産業上の利用分野〕

本発明は計算機システムにおけるタスクスケジュール方式に関し、特に複数のプロセッサが主メモリを共用し、複数の処理（タスク）を並行して実行する高多重結合プロセッサシステムにおける高性能タスクスケジュール方式に関する。

〔従来の技術〕

従来、タスクスケジュール方式については、先入れ先出し方式、プライオリティによる方式、プロセッサ使用時間の長いタスクよりI/Oの多いタスクを優先する方式、処理時間の目標値を守るように処理順序を変える方式等があった。これらの方式では、タスクの切替えオーバーヘッドは、考慮されていなかった。

また、タスクの切替えオーバーヘッドを考慮したタスクスケジュール方式としては、例えば、特開昭63-113738号 公報に開示された方式が知られている。しかし、この方式においても、プロセッサ間で共用されているキャッシュメモリ等のハードウェアを考慮したタスクスケジュール方式は配慮されていない。

特定のタスクを特定のハードウェアリソースが利用可能なプロセッサ（またはそのグループ）でのみ実行させるタスクスケジュール方式として、特開昭62-75739号公報（米国特許4808167に対応）に開示された方式が知られている。この方式では、タスク毎に設けられたタスクテーブル

がなされておらず、プロセッサを限定して処理を依頼しようとしても、すべてのプロセッサが実行可能なタスクのキュー上のすべてのタスクをサーチして実行するタスクを選択しているため、指定外のプロセッサがキュー上で当該タスクの実行可能性的チェックをして、スキップするオーバーヘッドが大きいという問題もあった。以下、これについて具体的に説明する。

第6図は、従来のタスクスケジュール方式における、プロセッサ間で共用されているキャッシュメモリ等のハードウェアを考慮したシステム構成例を示す図である。このシステムにおいては、主メモリ1aを共用するプロセッサ2a, 2b, 2cおよび2dは、各々独立にタスクを処理することができる。プロセッサ2aと2b, 2cと2dはそれぞれ、キャッシュメモリ1b, 1c, 1dを共用する。また、システムテーブル5およびスケジュールロックエントリ7を主メモリ1a上に持つ。

主メモリ1a上には、タスク情報を管理するた

特開平3-218534 (S)

内に当該タスクを実行可能なプロセッサをビットマップとして設定する。タスクスケジュール時に当該ビットマップをチェックし、許可されたタスクのみをスケジュールすることにより、上記のプロセッサ限定実行を実現している。

（発明が解決しようとする課題）

前記各従来技術は、タスクを実行中のプロセッサが、その時点での最も優先度の高いタスクを実行するように優先順位により配列された実行可能なタスクのキューをサーチし、タスクを切替えながら動作する。従って、前回タスクを実行した際のデータがキャッシュメモリに残っており、これを再利用することにより、キャッシュミスは低減し、MIPSを向上させることが可能な場合について配慮がなされておらず、当該キャッシュメモリを利用できないプロセッサにおいて、タスクの実行を開始してしまい、このため、余分なキャッシュミスが発生するという問題があった。

また、特定のプロセッサでのみ実行可能なタスクのスケジュールオーバーヘッドの削減に関する配

めの領域（タスクテーブル）10a, 10f, 10g, 10h, 10iが、タスク対応に設けられており、同時に、タスク間の処理の独立性を高めるために使用される多重アドレス空間を管理するための領域（空間テーブル）14p, 14q, 14r, 14sが空間対応に設けられている。この空間テーブルのうち、実行可能なタスクテーブルをチェインしているものは、空間の優先順位15p, 15q, 15sに従って、システムテーブル5からチェインされている。

ここでは、タスクテーブル10hに対応するタスク（以下、単に「タスク10h」という）は、プロセッサ2a, 2bにのみ実装されている内蔵アレープロセッサ（以下、「IAP」という）を使用するため、プロセッサ2a, 2bでのみ実行可能であることを示す印（フラグ）が付けられている。

第8図において、プロセッサ2aがタスクスケジュールを行う場合、先頭の空間テーブル14pの先頭のタスク10aが選択され実行される。こ

特開平3-218534 (4)

のとき、タスクテーブル10cおよび実行可能なタスクがなくなった空間テーブル10eおよび実行可能なタスクがなくなった空間テーブル14pは、それぞれのキューから外されるのが普通である。

更に、プロセッサ2b、2cがタスクスケジュールを行うと、それぞれ、タスク10f、10gが選択され、実行される。次に、プロセッサ2dがタスクスケジュールを行うと、選択されるタスクは順序から言えばタスク10hであるが、前述の如くタスク10hには、プロセッサ2a、2bでのみ実行可能であることを示す印（フラグ）が付けられているので、これを検知してスキップし、次のタスク10iが選択され実行される。なお、タスクテーブル、空間テーブルのサーチと操作を行う際のプロセッサ間の排他を行うため、タスクスケジュールの間、スケジュールロックエントリがロックされる。

前述の如く、上記技術によれば、特定のプロセッサでのみ実行可能なタスクのスケジュールオー

バヘッドの削減に関する配慮がなされていないため、プロセッサを限定して処理を依頼しようとしても、すべてのプロセッサが実行可能なタスクのキュー上のすべてのタスクをサーチして実行するタスクを選択しているため、指定外のプロセッサがキュー上で当該タスクの実行可能性のチェックをすることにより、スキップするオーバーヘッドが大きくなるという問題があった。

また、タスクに対応するタスクテーブル毎に実行可能なプロセッサとの対応関係を設定しており、この対応関係を変更する必要がある生じた場合、タスク数が多いと、変更オーバーヘッドが大きくなるという問題があった。

本発明は上記事情に鑑みてなされたもので、その目的とするところは、従来の技術における上述の如き問題を解消し、上述のプロセッサ対応のハードウェアの効率的な利用を図り、更に、オーバーヘッドの少ないプロセッサ限定タスクスケジュールを可能とするタスクスケジュール方式を提供することにある。

〔問題を解決するための手段〕

このような目的を達成するために、本発明は、複数のプロセッサが主メモリを共用し、複数の処理（タスク）を並行して実行する高多重連結プロセッサシステムにおいて、特定のプロセッサ（またはそのグループ）のみから利用可能なハードウェアが存在する場合に、仮想的なプロセッサを設けて、前述の特定のプロセッサおよびタスクを前記仮想的なプロセッサに対応付けるとともに、下記のいずれかを実行することを特徴とする。

- (1) 前記仮想的なプロセッサに対応して実行待ちタスクのキューを設けて、前述のプロセッサは、対応する前記仮想的なプロセッサのキューからタスクを取出し、タスクの実行を開始する。
- (2) 各タスク毎に対応する仮想的なプロセッサを表示したタスクテーブルを設け、前記プロセッサは、各タスクのタスクテーブルにより自プロセッサに対応する仮想的なプロセッサに対応付けられたタスクを選択し、実行を開始する。

〔作用〕

本発明に係るタスクスケジュール方式においては、特定のプロセッサ（またはそのグループ）のみから利用可能なハードウェアが存在する場合に、仮想的なプロセッサを設けて、上述の特定のプロセッサおよびタスクを上記仮想的なプロセッサに対応付けるとともに上記仮想的なプロセッサに対応して実行待ちタスクのキューを設けて、前述の特定のプロセッサは、対応する上記仮想的なプロセッサのキューからタスクを取出し、タスクの実行を開始するようにしたので、例えば、キャッシュメモリ上に前回実行時のデータが残っている場合に、その再利用の可能性を向上させることができる。更に、各プロセッサは、当該プロセッサで処理不可能なタスクのサーチを行わず、実行可能なタスクのサーチのみを行うことにより、タスクサーチオーバーヘッドの削減を実現している。

また、別のタスクサーチオーバーヘッド削減策として、タスクスケジュール時に、仮想的なプロセ

特開平3-218534(5)

ッサに対応する仕事アドレス空間にある場合にのみアドレス空間内のタスクのサーチを実行し、空間レベルでのスキップを行うことにより、各プロセッサは当該プロセッサで処理不可能なタスクのサーチを最小限とする方策を実現している。

また、プロセッサあるいはハードウェアリソースの故障等によりタスクとプロセッサの対応付けを変更する必要が生じた場合、プロセッサと仮想的なプロセッサとの対応付けを変更すれば良い。一般的に、タスク数よりはタスクを対応付ける対象となるハードウェアリソース数は少ない。従って、対応関係の変更はタスク対応のテーブル内にプロセッサとの対応付けを保持している場合と比較してより少ないオーバーヘッドで実現可能である。

【実施例】

以下、本発明の一実施例を図面を用いて具体的に説明する。

第2図は、本発明の一実施例に係るタスクスケジュール方式におけるシステム構成を示す図である。主メモリ1を共用するプロセッサ2a、2b、

2cおよび2dは、各々独立にタスクを処理することができる。プロセッサ2aと2b、2cと2dは、それぞれキャッシュメモリ19a、b、19c、dを共用し、また、システムテーブル5およびスケジュールロックエントリ7を主メモリ1上に持つ点は先に示したシステムと同様である。

上記主メモリ1上には、プロセッサ2a、2b、2cおよび2dにそれぞれ対応して、1Pテーブル3a、3b、3cおよび3dが設けられている。また、重複を許すプロセッサグループに対応する仮想プロセッサ(VIP)に対応するテーブル(以下、「VIPテーブル」という)8a、8b、8c、8dが設けられている。なお、本実施例においては、VIPテーブル8aはプロセッサ2a、2bにのみ実装されている前記IAPおよびキャッシュメモリ19a、bに対応し、VIPテーブル8cはキャッシュメモリ19c,dに対応している。

また、主メモリ1上には、タスク情報を管理するための領域(以下、「タスクテーブル」という)10e、10f、10g、10h、10iが、タ

スク対応に設けられている。タスクテーブル10e~10iには、それぞれ、VIP番号12e~12iおよびVIP候補番号20e~20iを予め設定する。更に、タスクテーブル10e~10iをアドレス空間対応にVIPテーブルにチェインするための制御ブロックとして、SQ(スケジュールキュー)テーブル16p~16sを用意する。なお、各アドレス空間の全タスクが、各々、同一のVIPテーブルからチェインする方式も可能である。

なお、ここでも、タスク10eは、優先タスクとして、VIP候補番号20eおよびVIP番号12eには1のみを設定する。同様に、IAPを使用するタスク10hに対しては、2のみを設定する。VIPテーブル3a、3bのVIPリスト4a、4bには、VIPテーブル8k、8lへのポインタをこの順序で設定する。また、IPテーブル3c、3dには、VIPテーブル8mへのポインタを設定する。第3図は本実施例の方式におけるタスクテーブルと空間テーブル、SQテーブルの関係を示す図である。タスクテーブルから空間テーブルを辿ることができる。また、空間テーブルから、VIP番号毎に対応するSQテーブルを辿ることができる。

本実施例におけるプロセッサによるタスクスケジュールの処理の流れを第1図に示す。まず、ステップ601で、自プロセッサに対するIPテーブルを見つける。そして、IPテーブルのVIPリストから最初のVIPを求める(ステップ602)。次に、VIPテーブル中のロックエントリをロックし(ステップ603)、VIPタスクスタックに

タスクテーブルがスタックされているかチェックする(ステップ604)。スタックされている場合は、VIPタスクスタックから全タスクテーブルをCS命令により外す(ステップ605)。タスクテーブルを1つとり(ステップ606)、TSKENQ(Task Enqueue)ルーチンをコールして、タスクテーブルをVIPのキューに入れる(ステップ607)。未処理のタスクテーブルがまだあるなら(ステップ608)、更に、TSKENQを繰り返す。

VIPテーブルにSQテーブルがチェーンされていない場合(ステップ609)には、VIPテーブルのロックエントリのロックを外す(ステップ610)。IPテーブルのVIPリストに、次のVIPテーブルが含まれているならばプロセッサはWAITに入る。VIPにSQテーブルがチェーンされている場合、先頭のSQテーブルを求め(ステップ612)、SQテーブルにチェーンされているタスクテーブルの先頭のを求める(ステップ613)。TSKDEQ(Task Dequeue)

TSKDEQルーチンの処理の流れを示すものである。まず、タスクテーブルをSQテーブルのチェーンから外す(ステップ101)。SQテーブルにタスクテーブルがなくなった場合(ステップ102)には、SQテーブルを、VIPテーブルのチェーンから外す(ステップ4103)。

上記実施例によれば、実行中断したタスクが、再度スケジュールされる際、同一のキャッシュメモリを有するプロセッサで実行される可能性が高くなり、また、直接このタスクを実行開始できるプロセッサ以外は、当該タスクをサーチせず、タスクスケジュールにおけるサーチオーバーヘッドを削減する効果もある。

次に、本発明の他の実施例を、第4図に示すテーブル関連図および第8図の処理の流れ図により説明する。第4図に示す如く、例えば、IPテーブル30には高負荷時のVIPリスト210、および低負荷時のVIPリスト220が予め設定されている。負荷変動の監視処理は、第8図に示す如く、定期的に起動され、別に設定されている手

特開平3-218534(6)

ルーチンをコールして、タスクテーブルをSQテーブルのチェーンから外す(ステップ614)。VIPテーブルのロックエントリのロックを外し(ステップ615)、対応するタスクの実行を開始する。

第9図は、上記ステップ607に示したTSKENQルーチンの処理の流れである。まず、第3図に例を示す如く、タスクテーブル10jからVIP番号12jを求め(ステップ81)、更に、空間テーブル14pから、VIP番号12jに対するSQテーブル16peを、SQテーブルリスト18pを使用して求める(ステップ92)。次に、タスクテーブルを、SQテーブルからのタスクテーブルのキューの末尾に挿入する(ステップ93)。SQテーブルが、まだVIPテーブルからチェーンされていない場合には(ステップ94)、SQテーブルを、VIPテーブルのチェーンに優先順位20pを考慮して挿入する(ステップ95)。

第10図は、前述のステップ614に示した

段により測定された前回該処理を行ってからのCPU使用率を使用して、CPU使用率を求める(ステップ81)。CPU使用率が予め与えられた限界値より大きい場合(ステップ82)、全IPテーブル中のVIPを高負荷時のVIPに従って変更する(ステップ83)。IPテーブル30においては、VIPリスト40としてVIPテーブル8mのみを設定する。第4図の設定内容に対する高負荷時のIPテーブルとVIPテーブルの対応関係は、第2図の通りである。また、CPU使用率が限界値より低い場合、全IPテーブル中のVIPリストを低負荷時のVIPリストに従って変更する(ステップ84)。IPテーブル30においては、VIPリスト40として、VIPテーブル8kおよびVIPテーブル8mを設定する。第4図は、低負荷時のIPテーブルとVIPテーブルの対応関係を示している。

上記実施例によれば、VIPとプロセッサとの対応を、プロセッサの負荷状態により変更することができ、処理の効率向上が可能になるという効

特開平3-218534(7)

果がある。

次に、プロセッサシステムの操作を行うための端末から入力された指示により、上記プロセッサとVIPとの関係およびVIPとタスクとの関係を端末に出力する例を示す。第4図のVIPテーブル8kに示す如く、テーブル中には、当該VIPに対応するタスクの数24kと、変更不可タスクの数25kが設定されている。このカウントは、タスクとVIPの対応を変更する都度、更新する。例えば、第2図に示したタスクテーブル10fは、VIP候補番号20fが2および3の複数となっており、変更可能である。これに対して、タスクテーブル10hは、VIP候補番号20hが2のみであり、変更不可タスクである。従って、VIPテーブル8kのタスク数24kは2であり、変更不可タスク数は1である。

第5図に示したVIPとタスクの関連表示は、このVIPテーブル中のタスク数および変更不可タスク数による。また、IPとVIPの関連表示は、IPテーブル中のVIPリスト、高負荷時の

リ110p~110sを設定する。また、各SQテーブル対応に、SQMSG(SQ Message)スタックを用意する。第11図では、SQテーブル16qgのSQMSGスタックにMSGテーブル111wが入れられている。

更に、システムテーブル5からチェーンされたMSGテーブルのチェーンが未使用MSGプールとして追加されている。第11図では、未使用MSGプールに、MSGテーブル111s, 111t, 111uが用意されている。第11図ではタスクテーブル中のVIP候補番号を省略しているが、本実施例でも同様に設定されているものとする。また、本実施例におけるタスクテーブルと空間テーブル、SQテーブルの関係は、先の実施例の場合と同様である(第8図参照)。

本実施例における、タスクが実行可能となったときの処理の流れを、第13図に示す。まず、未使用MSGプールから1つ、MSGテーブルを外す(ステップ131)。このMSGテーブルにタスクテーブルのアドレスを設定する(ステップ

VIPリストおよび低負荷時のVIPリストを使用して行う。

上記実施例によれば、必要に応じて、プロセッサシステムの操作を行うための端末から指示を行うことにより、プロセッサとVIPとの関係およびVIPとタスクとの関係を端末に出力させ、状態を確認することができる。

次に、本発明の更に別の実施例を示す。本実施例は、前述の実施例のタスクテーブル操作および排他制御方式を変更し、ロックの競合確率を低下させるものである。

第11図は、本実施例の方式におけるシステム構成図であり、以下の点を除いては、第2図に示した実施例と同様である。

VIPテーブル8k, 8l, 8mの各々に対応して、下記のVIPMSG(VIP Message)スタックを設ける。第11図においては、VIPテーブル8mのVIPMSGスタックにMSG(Message)テーブル111vが入れられている。SQテーブル16p~16sには、ロックエント

リ132)。タスクテーブルのVIP番号から、対応するVIPを求め(ステップ133)、VIPテーブルのMSGスタックにCS命令により、前記MSGテーブルを挿入する(ステップ134)。第11図には、VIPテーブル8mにMSGテーブル111vをスタックした例を示した。

本実施例におけるプロセッサによるタスクスケジュールの処理の流れを、第12図に示す。まず、自プロセッサに対するIPテーブルを見つける(ステップ1201)。IPテーブルのVIPから、最初のVIPを求める(ステップ1202)。ステップ1203では、VIPテーブル中のロックエントリをロックし、VIPMSGスタックにMSGテーブルがスタックされているか否かをチェックする(ステップ1204)。スタックされている場合は、VIPMSGスタックから、全MSGテーブルをCS命令により外す(ステップ1205)。

MSGテーブルを1つとり(ステップ1206)、SQENQ(SQ Enqueue)ルーチンをコールし

特開平3-218534(8)

て、MSGテーブルをSQテーブルのSQMSGスタックに入れるとともに、SQテーブルをVIPテーブルのキューに入れる(ステップ1207)。未処理のMSGテーブルがまだあるなら(ステップ1208)、更にSQENQを繰り返す。VIPテーブルに、SQテーブルがチェインしていない場合(ステップ1209)には、VIPテーブルのロックエントリのロックを外す(ステップ1210)。なお、IPテーブルのVIPリストに、次のVIPテーブルが含まれているなら(ステップ1211)、当該VIPテーブルに対して、上述の処理を繰り返す。VIPテーブルが含まれていない場合は、プロセッサはWAITに入る。

VIPテーブルにSQテーブルがチェインされている場合、操作対象としてチェイン先頭のSQテーブルを求め(ステップ1212)、SQMSGスタックにMSGテーブルあり、または、SQテーブルにタスクテーブルのチェインあり、をチェックする(ステップ1213)。どちらかが有の場合には、VIPテーブルのロックエントリの

ロックを外し(ステップ1214)、SQテーブルのロックエントリのロックを確保し(ステップ1215)、ASSCH(Address Space Schedule)ルーチンをコールして、空間に関するスケジュールを行う(ステップ1216)。

次に、SQテーブルのロックエントリのロックを外し(ステップ1217)、実行開始するタスクが見つかったか否かチェックする(ステップ1218)。見つかったときは対応するタスクの実行を開始する。見つからないときはVIPテーブルのロックエントリを再度ロックし(ステップ1219)、SQテーブルが現在VIPのSQテーブルチェインに入っているか否かチェックする(ステップ1220)。SQテーブルがチェインから外されている場合には、ステップ1209へ戻り、他のSQテーブルをさがす。また、SQテーブルがあるときは、SQテーブルをVIPのSQテーブルチェインから外す(ステップ1221)。更に、次のSQテーブルを求めて、あればその処理を行い(ステップ1210)、な

ければ、VIPのロックエントリのロックを外して(ステップ1218)、次のVIPの処理へ移る。

第14図は、SQENQルーチンの処理の流れを示すものである。まず、MSGテーブルからタスクテーブルを辿り、そこからVIP番号を求めて(ステップ141)、更に、空間テーブルからVIP番号に対するSQテーブルを、SQテーブルリストを使って求める(ステップ142)、MSGテーブルをSQテーブルからのSQMSGスタックに挿入する(ステップ143)。なお、SQテーブルがまだVIPテーブルからチェインされていない場合(ステップ144)には、SQテーブルを、VIPテーブルのチェインに、優先順位を考慮して挿入する(ステップ145)。

第15図に、空間内のタスク処理を行うASSCHルーチンの処理の流れを示す。まず、SQMSGスタックに、MSGテーブルがあるか否かチェックする(ステップ151)。MSGテーブルがあるとき、すべてのMSGテーブルをSQM

SGスタックから外す(ステップ152)、MSGテーブルを1つとり(ステップ153)、当該MSGテーブルに対応するタスクテーブルが、既にSQテーブルからチェインされているか否かチェックする(ステップ154)。チェインされていないときは、タスクテーブルをSQテーブルのタスクテーブルキューの末尾に挿入する(ステップ155)。次に、MSGテーブルを、未使用MSGテーブルに戻す(ステップ156)、これを、SQMSGスタックから外したMSGテーブルの、すべてに関して行う(ステップ157)。最後に、実行可能なタスクのタスクテーブルが、SQテーブルからチェインされているか否かチェックし(ステップ158)、ステップ159では、その先頭のタスクテーブルを、実行開始のためSQテーブルのタスクテーブルキューから外す。

次に、本発明の更に他の実施例を、第16図に示すシステム構成図を基に説明する。本実施例では、キャッシュメモリが二段階になっており、図に示す如く、第2図に示した実施例に対して、プ

特開平3-218534(9)

ロセッサ対応の第二のキャッシュメモリ161a~161dが追加されている。また、VIPテーブル8a~8dは各プロセッサ2a~2dに対応して設定されている。各プロセッサとVIPとの対応関係は、IPテーブル中のVIPリストにより、以下の如く設定されている。

各プロセッサは、まず、対応するVIPをチェックする。次に、当該IPとキャッシュメモリを共用するプロセッサに対応するVIPを、チェックする。例えば、プロセッサ2aは、まず、VIPテーブル8aをチェックする。これにより、キャッシュメモリ19abおよび第二のキャッシュメモリ161aのデータを再利用する確率を高める。このようなタスクが、VIPテーブル8aに存在しない場合、VIPテーブル8bをチェックすることにより、キャッシュメモリ19abのデータを再利用する確率を高める。その他の動作および処理の流れは、前述の実施例と同様である。

第17図は、本発明の更に他の実施例におけるシステム構成図である。

114
仕事有ビットマップ114f (VIP1に対するタスクであることを表す)をあらかじめ設定する。タスクテーブル10fを空間対応にシステムテーブル5にチェインするための制御ブロックとして、空間テーブル14aを用意する。他のタスクテーブルも同様である。タスクテーブル10aに対しては仕事有ビットマップ114aを設定する。IPテーブル3a, 3bのサービス対象VIPビットマップ31a, 3bのサービス対象VIPとの対応を表わすビットを設定する。また、IPテーブル3c, 3dには同様にしてVIP0, VIP2との対応を設定する。

各空間テーブルには、ロックエントリ110a、仕事有VIPビットマップ112a (左から順にVIP0, 1, 2に対応)、VIP対応カウンタ113a (上から順にVIP0, 1, 2に対応)が設けられている。

第18図は、本発明のタスクを実行可能な状態とした時の処理の流れを示す。まず、そのタスクに対応する空間テーブルを求める(ステップ

主メモリ1を共用するプロセッサ2a, 2b, 2c, 2dは、おのおの独立にタスクを処理することができる。プロセッサ2aと2b, 2cと2dはそれぞれキャッシュメモリ19abおよび19cdを共用する。システムテーブル5およびスケジュールロックエントリ7を主メモリ上に持つ点は前述した従来のシステムと同様である。

プロセッサ2a, 2b, 2c, 2dに対応してIPテーブル3a, 3b, 3c, 3dを設ける。また、IPテーブルには、VIPとの対応関係をビットにより表現したサービス対象VIPビットマップ31a, 31b, 31c, 31dが設けられている。ビットマップのビット0, 1, 2は、三種のVIP (VIP0, VIP1, VIP2)と対応を各々表している。ビットが1の時、そのIPはVIPに対応付けられている。

本発明の実施例では、VIP0は金IPで処理を行うタスクに、VIP1はキャッシュメモリ19abに対応し、VIP2はキャッシュメモリ19cdに対応する。タスクテーブル10fには

1701)。次に、その空間テーブルのロックエントリをロックし(ステップ1702)、タスクを空間テーブルのタスクキューに入れる(ステップ1703)。タスクテーブルの仕事有ビットマップにおいてオンとなっているビットに対応するVIPに対して、つぎの処理を行う(ステップ1704, 1707, 1707, 1708)。空間テーブルのVIP対応カウンタをプラスし、空間テーブルの仕事有VIPビットマップの対応するビットをオンに設定する。空間テーブルのロックエントリのロックを外し(ステップ1709)。この空間テーブルが空間テーブルキューに入っているかチェックしなければ、空間テーブルを空間テーブルキューに入れる(ステップ1712)。空間テーブルキューの操作の間、スケジュールロックエントリのロックを確保しておく(ステップ1711, 1713)。

本発明におけるプロセッサによるタスクスケジュールの処理の流れを第19図に示す。まず、自プロセッサに対するIPテーブルを見つける(ス

特開平3-218534 (10)

テップ1801)。IPテーブルからサービス対象VIPビットマップを求める(ステップ1802)。スケジュールロックエントリをロックし(ステップ1803)。空間テーブルキュー先頭の空間テーブルを求める(ステップ1804)。

システムテーブルに空間テーブルがチェインしていない場合(ステップ1805)、スケジュールロックエントリのロックを外し(ステップ1811)、プロセッサはウェートに入る。空間テーブルがチェインしている場合は、対象となった空間テーブルの仕事有VIPビットマップと前記サービス対象VIPビットマップの論理積を求め(ステップ1806)、結果がオールゼロかチェックする(ステップ1807)。結果がオールゼロの場合、前記空間テーブルの仕事有VIPビットマップがオールゼロかをチェックする(ステップ1808)。これもオールゼロの場合、対象空間テーブルを空間テーブルキューから外す(ステップ1809)。いずれの場合も次の空間テーブルを求め(ステップ1810)ステップ1805

へ戻る。

前述の結果がオールゼロでない場合、スケジュールロックエントリのロックを外し(ステップ1812)、空間テーブルのロックエントリをロックする(ステップ1813)。空間のタスクキュー先頭のタスクテーブルを求める(ステップ1814)。タスクテーブルのない場合、空間テーブルのロックエントリのロックを外し(ステップ1818)、スケジュールロックエントリをロックし(ステップ1819)。ステップ1810へ戻る。タスクテーブルのある場合、タスクテーブルの仕事有ビットマップと前述のサービス対象VIPビットマップの論理積を求め(ステップ1816)、結果がオールゼロなら次のタスクテーブルを求め(ステップ1817)。ステップ1815へ戻る。論理積がオールゼロでない場合、タスクキュー(TSKDEQ)ルーチンをコールして、タスクテーブルを空間テーブルのチェインから外す(ステップ1820)。空間テーブルのロックエントリのロックを外し(ステップ1821)。

対応するタスクの実行を開始する。

第20図にタスクキュー(TSKDEQ)ルーチンの処理の流れを示す。まず、タスクテーブルを空間テーブルのチェインから外す(ステップ1901)。タスクテーブルの仕事有ビットマップでオンとなっているビットに対応するVIPの各々に対して下記の処理を行う(ステップ1902、1906、1907)。空間テーブルのVIP対応の仕事有カウンタをマイナス1する(ステップ1908)。カウンタがゼロになったなら(ステップ1904)、空間テーブルの仕事有VIPビットマップの対応するビットをCS(Compare and Swap)命令によりリセットする(ステップ1905)。

第21図は、VIPがキャッシュメモリのように性能は低下するが他のVIPのもので代替可能なハードウェアリソースに対応して設定されている場合のプロセッサ故障時の処理の流れを示す。第17図に示すように、例えばIPテーブル3a、3bにはサービス対象VIPビットマップ31a、

3b(VIP0およびVIP1を指定)があらかじめ設定されている。

そして、例えばプロセッサ2aの故障時にプロセッサ故障時の処理が起動され、第21図に示すように、故障したプロセッサのサービス対象VIP(ここではVIP0およびVIP1)を求める(ステップ2001)。各VIPに対して下記の処理を行う(ステップ2003、2007)。正常に動作中のプロセッサのIPテーブルをサーチし(ステップ2002、2004、2005)、前記VIPに対するサービス対象VIPビットマップがオフのIPテーブルを見つけたら当ビットをオンに渡える(ステップ2003、2008)。この例では、VIP0に対しては、これをサービスしないプロセッサはないため、テーブルの番替は発生しない。しかし、VIP1に対してはIPテーブル3aのサービス対象VIPビットマップ31aにおいて、VIP1に対応するビットが設定され、「111」となる。

次にプロセッサ3bが故障となった場合、同様

時間平3-218534 (11)

の処理によりIPテーブル3dのサービス対象VIPビットマップ31dにおいて、VIP1に対応するビットが設定される。これにより、VIP1をサービスするプロセッサがなくなることなく、処理を継続することが可能である。

第22図は、プロセッサシステムの操作を行うための端末から入力された指示により、空間とVIPタスクの関連を端末に表示する例である。第17図の空間テーブル14qに示すように、テーブル中には当該空間に対応する各VIP対応のタスクの数113qが設定されている。このカウンタは第18図および図20図の処理フローにより示した通り、空間のタスクキューのタスクを出し入れする都度、更新する。例えば第17図において、空間テーブル14qはVIP0に対するタスクが1個、VIP1に対するタスクが1個そのタスクキュー中に存在する。これに対して、空間テーブル14rはVIP1のタスクが1個のみである。第22図に示す空間とVIPタスクの関連表示は、この空間テーブル中のカウンタによる。こ

およびタスクを前記仮想的なプロセッサに対応付けるとともに(1)前記仮想的なプロセッサに対応して実行待ちタスクのキューを設けて、前述のプロセッサは、対応する前記仮想的なプロセッサのキューからタスクを取出し、タスクの実行を開始する。あるいは、(2)タスクおよび空間毎に対応する仮想的なプロセッサのビットマップを設けたことにより、空間レベルでビットがオンとなっている仮想的なプロセッサに対応するプロセッサ以外は当該空間内のタスクをサーチしないことにより、プロセッサ対応のハードウェアの効率的な利用を図り、更に、オーバーヘッドの少ないプロセッサ限定タスクスケジュールを可能とするタスクスケジュール方式を実現できるという顕著な効果を奏するものである。

また、仮想的なプロセッサを設けたことにより、タスクプロセッサの対応付けの度合いが低いオーバーヘッドで実現可能となっている。

4. 図面の簡単な説明

第1図は本発明の一実施例を示すタスクスケジ

の例では、空間テーブル14qは第一行目に、空間テーブル14rは第二行目に表示されている。また、各VIPに対するタスク数の合計が最終行に表示されている。

なお、前述の各実施例の説明においては、本発明を複数のプロセッサが主メモリを共用するとともに、複数のアドレス空間を有し、少なくとも前記各アドレス空間の一つで、複数の処理(タスク)を並行して実行する如く構成されたシステムに対して適用して例のみを示したが、本発明はこれに限定されるものではなく、単一のアドレス空間のみを有するシステムに対しても、有効に適用可能であることは言うまでもない。

【発明の効果】

以上、詳細に説明した如く、本発明によれば、複数のプロセッサが主メモリを共用し、複数の処理(タスク)を並行して実行する高多重密結合プロセッサシステムにおいて、特定のプロセッサのみから利用可能なハードウェアが存在する場合に、仮想的なプロセッサを設けて、前述のプロセッサ

ユール処理の流れを示すフローチャート、第2図は本発明の一実施例のシステム構成を示す図、第3図はタスクテーブル、空間テーブル、SQテーブルの関連を示す図、第4図はIPテーブルとVIPテーブルの関連を示す図、第5図はVIPとタスク、IPとVIPの関連表示例を示す図、第6図は従来方式によるシステム構成を示す図、第7図はタスクの実行可能化の処理の流れを示すフローチャート、第8図は負荷監視によるIPとVIPの関連切替処理の流れを示すフローチャート、第9図はTSKENQ処理の流れを示すフローチャート、第10図はTSKDEQ処理の流れを示すフローチャート、第11図は本発明の他の実施例のシステム構成を示す図、第12図はそのタスクスケジュール処理の流れを示すフローチャート、第13図はタスクの実行可能化の処理の流れを示すフローチャート、第14図はSQENQ処理の流れを示すフローチャート、第15図はASSCH処理の流れを示すフローチャート、第16図は本発明の更に他の実施例のシステム構成

特開平3-218534 (12)

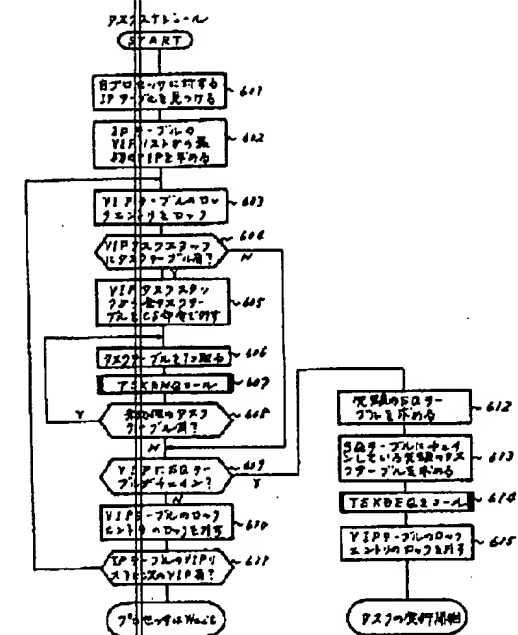
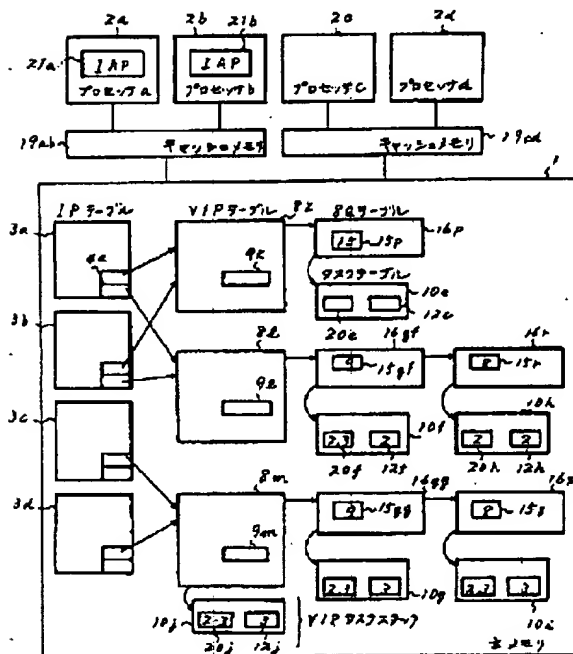
第 1 図

を示す図。第 17 図は、本発明のさらに他の実施例のシステム構成を示す図。第 18 図は、タスクを実行可能にするための処理の流れを示すフローチャート。第 19 図は、タスクスケジューリングの処理の流れを示すフローチャート。第 20 図は、タスクデキューリングの処理の流れを示すフローチャート。第 21 図は、プロセッサ故障時の処理の流れを示すフローチャート。第 22 図は、空間とVIPタスクの関連表示例を示す図である。

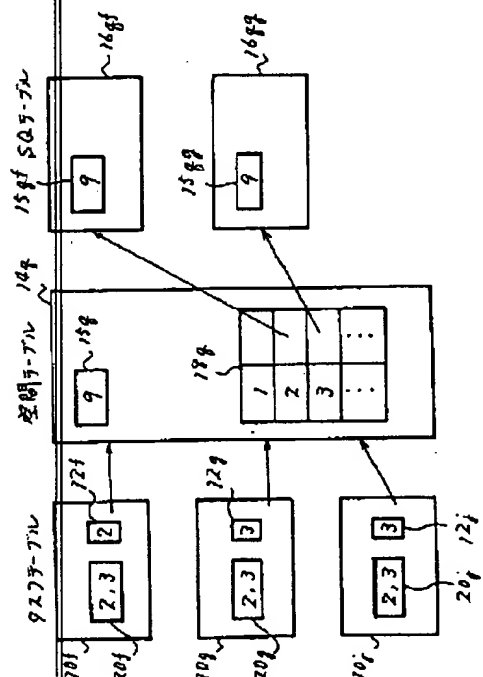
1: 主メモリ、2a~2d: プロセッサ、18a b, 18c d, 161a~161d: キャッシュメモリ、3a~3d: IPテーブル、8a~8d, 8k~8m: VIPテーブル、10e~10j: タスクテーブル、16p~16s: SQテーブル、14q, 14r, 14s: 空間テーブル、4a~4d: サービス対象VIPビットマップ、110q: ロックエントリ、112q: 仕事有VIPビットマップ、113q: VIP対応カウンタ。

代理人弁護士 小川 勝 男

第 2 図

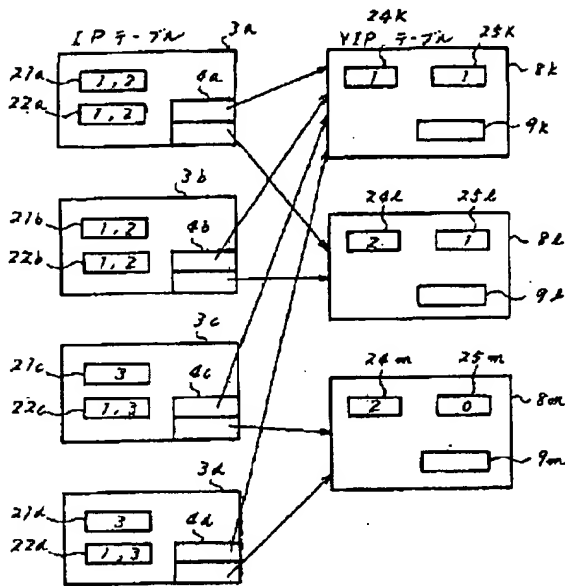


第 3 図



特開平3-218534 (13)

第 4 図

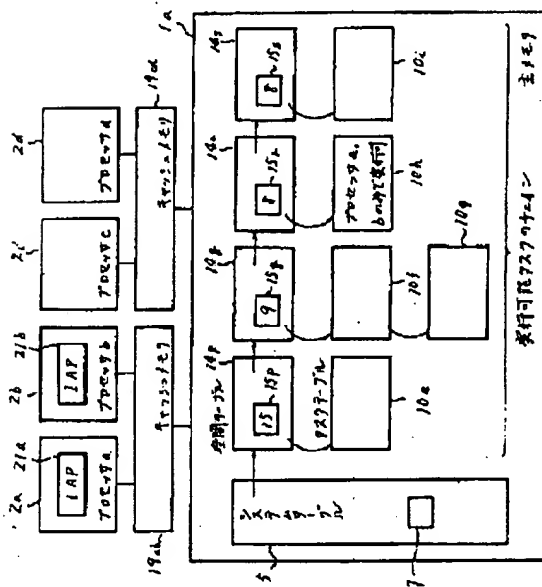


第 5 図

VIP-TASK COUNT DISPLAY			
VIP#	TASK COUNT	AFFINITY	TASK COUNT
2	2	1	0
3	2	2	0

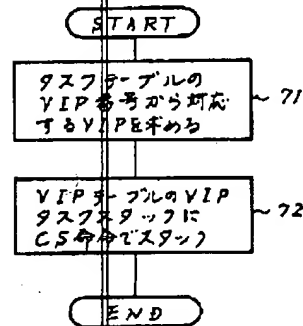
VIP-TASK LIST DISPLAY			
VIP#	TASK LIST	HIGH LOAD	LOW LOAD
1	12	12	12
2	12	12	12
3	3	3	13
4	3	3	13

第 6 図



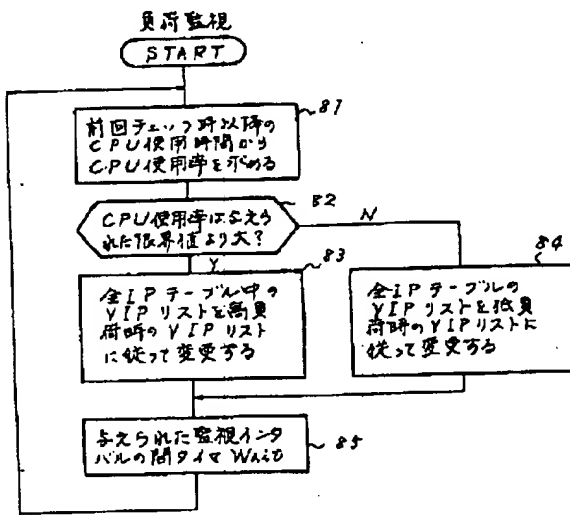
第 7 図

タスクの実行可能とした時の処理

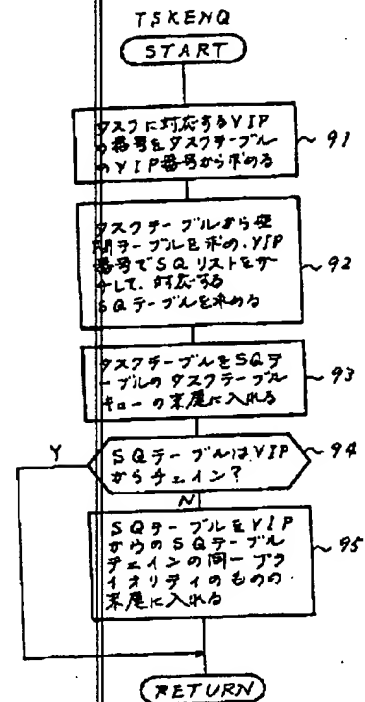


特開平3-218534 (14)

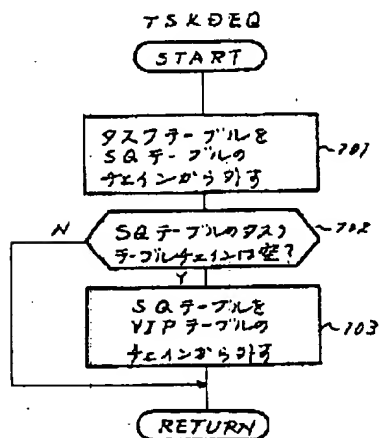
第 8 図



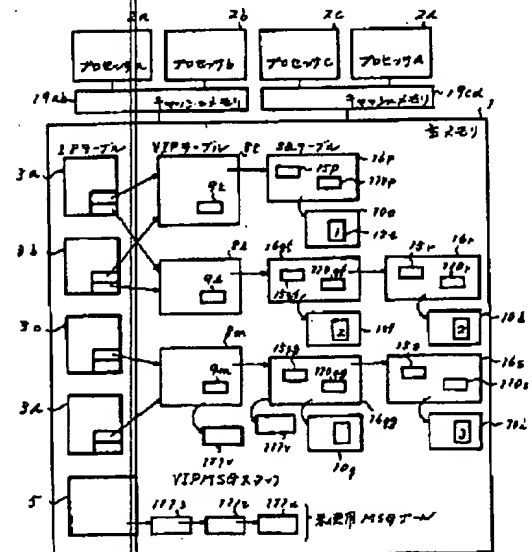
第 9 図



第 10 図

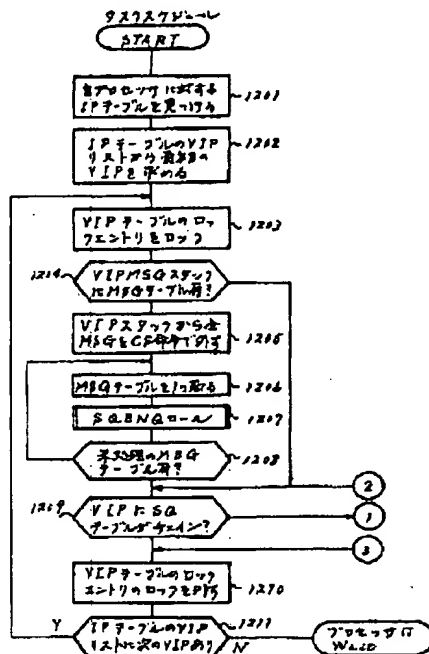


第 11 図

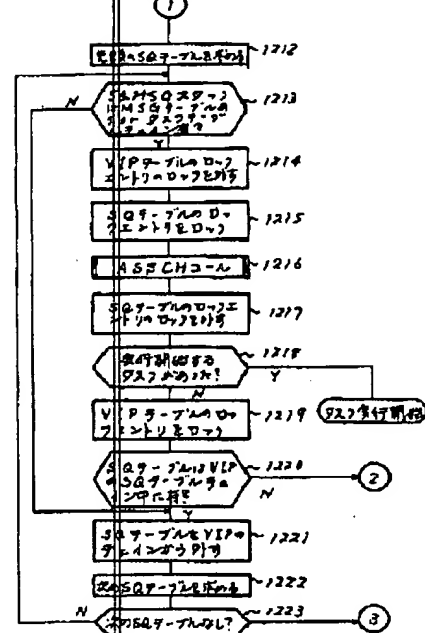


特開平3-218534 (16)

第 12 圖 (2)

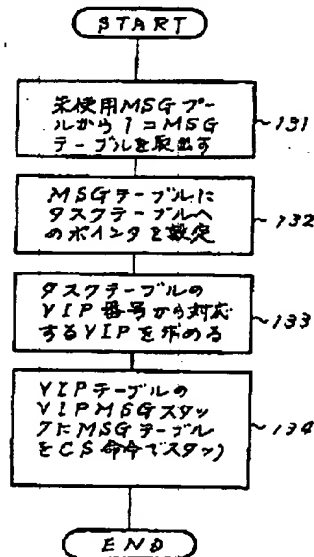


第 12 圖 (4)



第 八 回

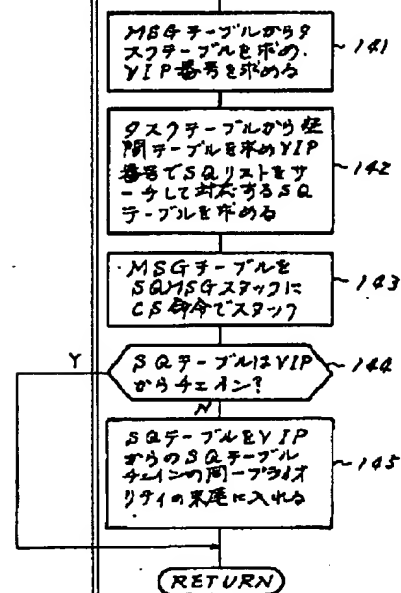
タスクを実行可能とした時の処理



第14圖

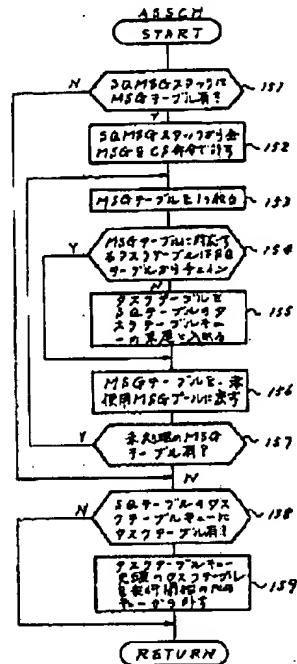
SAENA

START

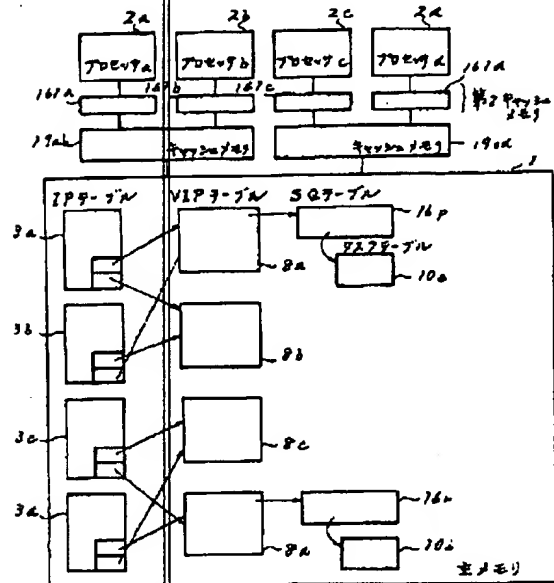


特開平3-218534 (18)

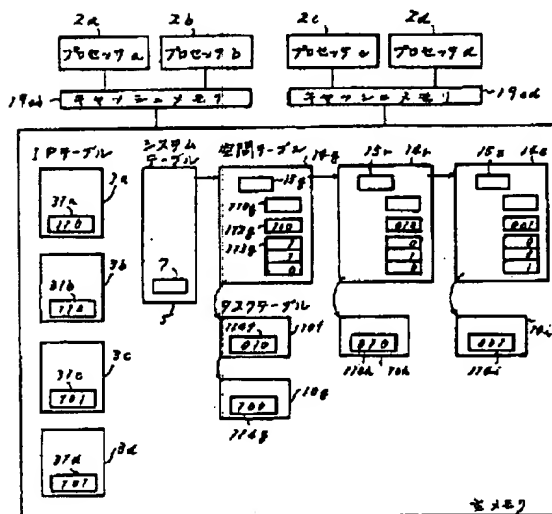
第 15 圖



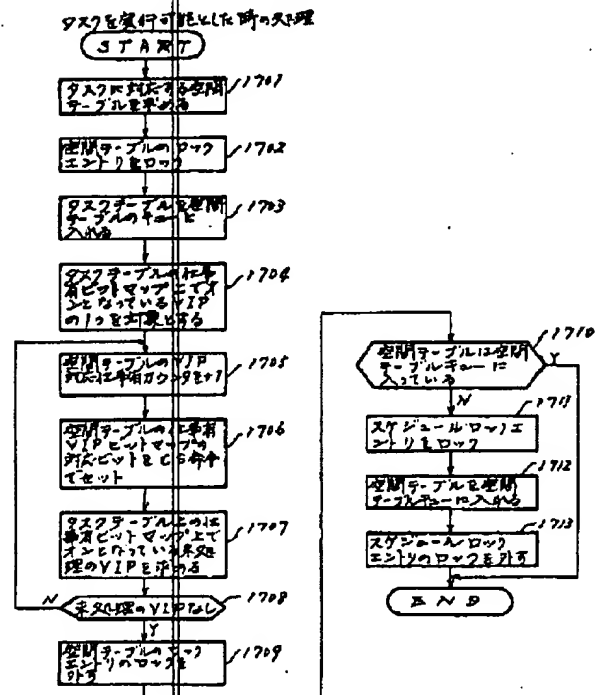
第 16 回



第 17 圖



第 18 圖



特開平3-218534(18)

第1頁の続き

優先権主張

⑫発明者

⑬発明者

⑫平1(1989)11月21日⑬日本(JP)⑭特願 平1-304236

佐藤

一 浩

神奈川県横浜市戸塚区戸塚町5030番地 株式会社日立製作
所ソフトウェア工場内

原 田

晃

神奈川県横浜市戸塚区戸塚町5030番地 株式会社日立製作
所ソフトウェア工場内

This Page is inserted by IFW Indexing and Scanning
Operations and is not part of the Official Record

BEST AVAILABLE IMAGES

Defective images within this document are accurate representations of the
original documents submitted by the applicant.

Defects in the images include but are not limited to the items checked:

- ☒ BLACK BORDERS
- ☒ IMAGE CUT OFF AT TOP, BOTTOM OR SIDES
- ☐ FADED TEXT OR DRAWING
- ☒ BLURED OR ILLEGIBLE TEXT OR DRAWING
- ☐ SKEWED/SLANTED IMAGES
- ☒ COLORED OR BLACK AND WHITE PHOTOGRAPHS
- ☒ GRAY SCALE DOCUMENTS
- ☐ LINES OR MARKS ON ORIGINAL DOCUMENT
- ☐ REPERENCE(S) OR EXHIBIT(S) SUBMITTED ARE POOR QUALITY
- ☐ OTHER: _____

IMAGES ARE BEST AVAILABLE COPY.

As rescanning documents *will not* correct images
problems checked, please do not report the
problems to the IFW Image Problem Mailbox